# Ozobot Editor—**Lesson: Array Primer**—Student Guide

*by Richard Born, Associate Professor Emeritus, Northern Illinois University*

***Introduction***

This lesson will take you through a sequence of eleven short Ozobot Editor programs that bring arrays to life in a way that will help you understand this data structure. Through the use of a provided Ozomap, Ari or Evo demonstrate array elements by *speaking* the contents of the element while following a line with the array's indexes and stopping at the intersections, as shown in Figure 1.

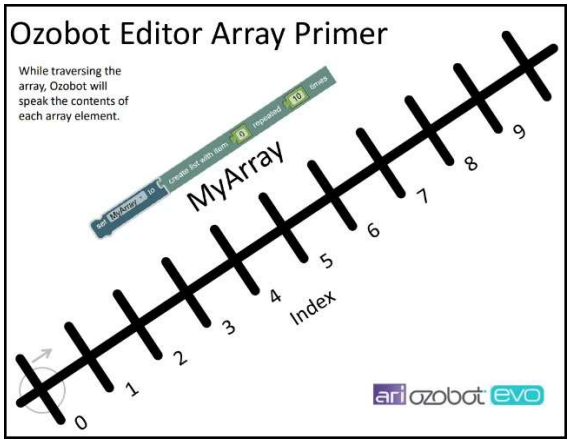


*Figure 1*

***What is an array?***

Before working with arrays in the Ozobot Editor, it might be a good idea to think of how the word *array* is used in everyday conversation. Here are some example sentences:

- There is a beautiful *array* of flowers in that vase.
- The *array* of solar panels on my neighbor's roof is soaking up the sun.
- Mark has an *array* of baseball bats in the corner of his bedroom.
- You can choose from an *array* of colors to paint your house.
- Sarah has an *array* of dresses in her closet.

These sentences all share a common thread. Each sentence references a *group of identical or similar things with the same name*: flowers, solar panels, bats, colors, or dresses. The concept of an array in computer science also shows this common feature. An array in computer science is *a group of sequential memory locations for storing similar information*. Examples include arrays of:

- Names of customers
- Restaurants within fifteen miles of your house
- Prices for items that you purchase while visiting the grocery store
- Colors that Ozobot encounters while following a line
- Directions that Ozobot takes for each step in a random walk

The name of the array for the "storing colors" example above might be *color*. Individual elements in an array are identified by an *index*. The first element in an array has an index 0. Languages such as JavaScript place the

index in square brackets. So, we could identify the first three elements of the array *color* by *color[0]*, *color[1]*, and *color[2]*. Array elements in the Ozobot Editor can hold any integer from -199 through 199. If we let 1 represent the color red, 2 green, and 3 blue, then the contents of *color[0]*, *color[1]*, and *color[2]* might be 2, 3, and 1 if Ozobot encountered the colors green, blue, and red in that order.

A "*set variable to"* block names an array and a "*create list"* block in the Ozobot Editor workspace is used to specify the length of an array and fill the array with initial default values. The minimum length of an array is 1 item, the maximum 128 items, with indexes ranging from 0 to 127. You can have more than one array, if needed, in your program logic.

***Array Declaration***

Most of the short programs that we study in this *Ozobot Editor Array Primer* have a single array of length 10 whose name is *ArrayA* and with indexes varying from 0 through 9. See Figure 2 for example array declaration blocks, and a view of the contents of each of the memory locations in the array.


| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 2*

The indexes are shown in square brackets [ ]. Note that arrays are indexed starting at 0, so the last index in the array is always 1 less than the declared length of the array. In ArrayA for these programs, the length is 10 and the last index is therefore 9. It should be noted that in the event the index is lower than 0 or higher than *the array length minus 1*, the program will terminate with an *index out of bound* error.

With the discussion of array declaration complete, we can now begin looking at each of our short Ozobot Editor programs, in which Evo is used to demonstrate some simple array manipulation concepts. You are encouraged to build them in the editor and then run them using the provided Ozomap. Note that Ozobot should initially be placed in the gray circle facing the direction shown by the arrow shown in Figure 1.

***Program 1: Zilch, Zip, Nada, Nothing***

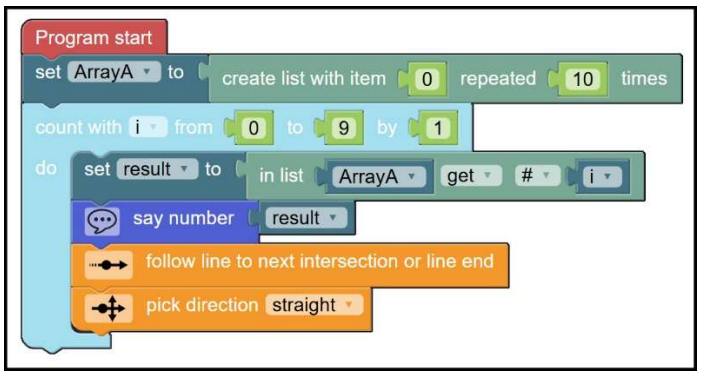

*Figure 3*

After the array declaration, we set up a loop in which the index (*i*) into the array varies from 0 through 9. Within the loop, we make use of the *"in list **get** #"* block to get the *value* of the array element whose index # is *i*, storing the value in the variable result. Then Evo speaks the *result* and follows the line to the next intersection. Fill in the values below that Evo speaks while the program is running.

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 2: All the Same to Me***

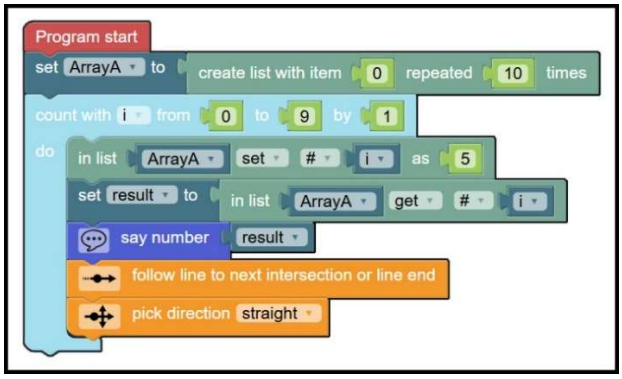


*Figure 4*

The program shown in Figure 4 introduces the "*in list **set** #*" block. For each index *i* from 0 through 9, the value 5 is written to the element with that index, wiping out the value 0 that was specified in the "create list" block. Other than that, the program is identical to the program of Figure 4. Fill in the values below that Evo speaks while the program is running.

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 3: To Err is Human (a.k.a., to really foul things up requires a computer)***

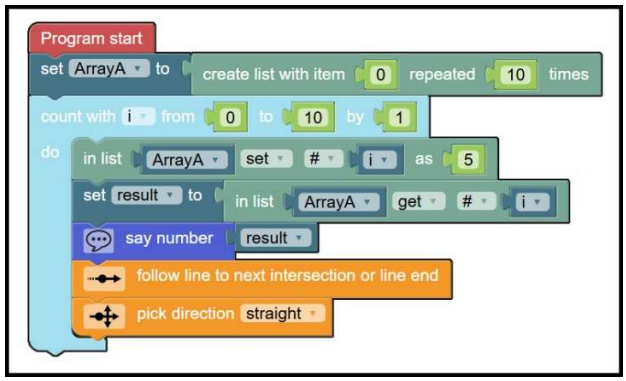


*Figure 5*

The program of Figure 5 has one small difference when compared to the program blocks of Figure 4. Can you find the difference? That's right—the loop goes from 0 to 10 rather than from 0 to 9. 10 is larger than the highest index allowed for an array of length 10. (Remember that the allowed indexes range from 0 to *length-1* for an array whose number of elements is *length*.) How does Ozobot respond to this when running the program on the Ozomap? Make sure you let the program run to its end.

*Ozobot's Response*:_____

***Program 4: The Count***

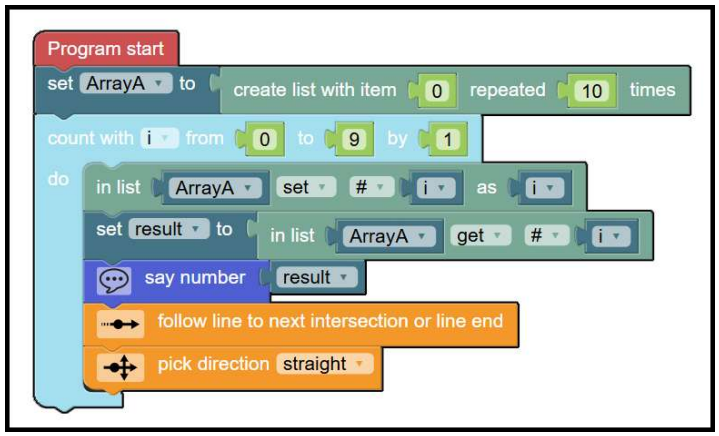


*Figure 6*

The program of Figure 6 has one small difference when compared to the program blocks of Figure 4. Can you find the difference? That's right—the *"in array … set element … to value …"* block has a *"to value"* with the variable *i* rather than the number 5. What do you then expect the value in each element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 5: Counting by 5's***

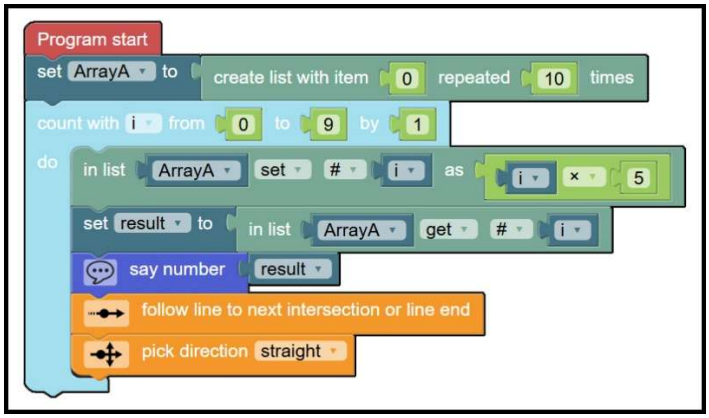


*Figure 7*

The program of Figure 7 differs from that of Figure 6 only in the *"to value"* argument of the *"set element"* block. Instead of the variable *i*, the *"to value"* is the arithmetic expression *i x 5*. What do you expect the value in each array element to be? Fill in the values below that Evo speaks while the program is running. Are you expectations correct?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 6: All Squared Up***

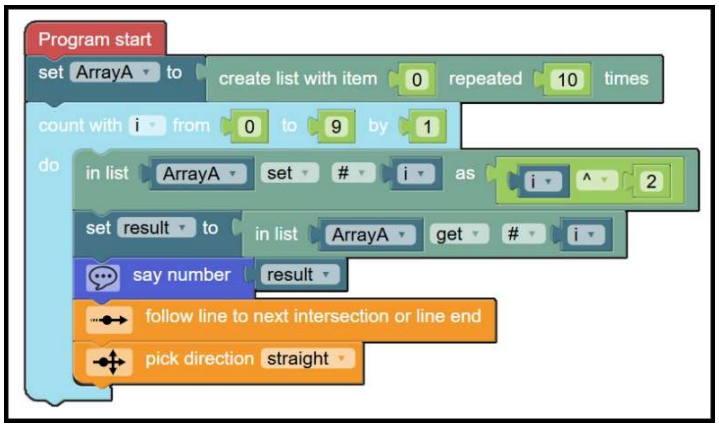


*Figure 8*

Study the program of Figure 8. What do you expect the value in each array element to be? Fill in the values below that Ozobot speaks while the program is running. Are you expectations correct?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 7: Blast off!***

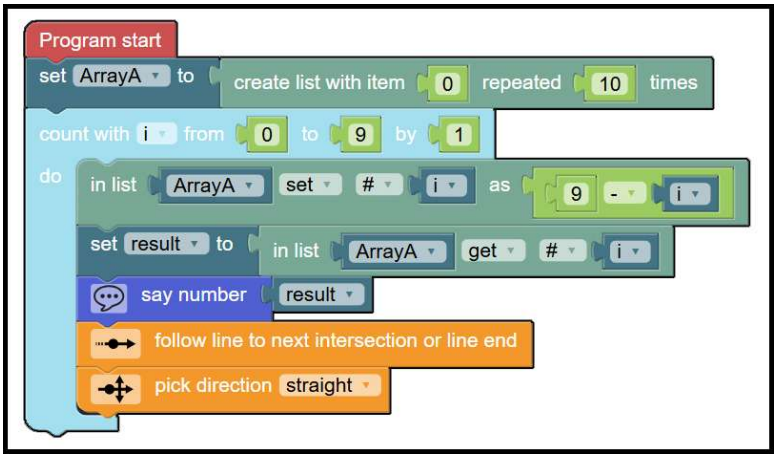


*Figure 9*

How does the program of Figure 9 differ from the program of Figure 8? What do you expect the value in each array element to be? Fill in the values below that Ozobot speaks while the program is running. Are you expectations correct?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 8: Getting Dicey***

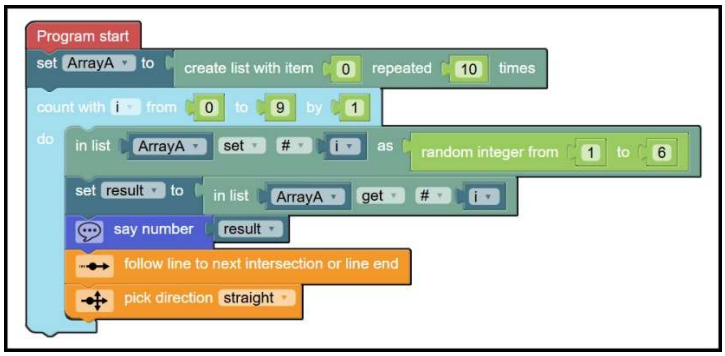


*Figure 10*

How does the program of Figure 10 differ from the program of Figure 9? What do you expect the values in each array element to be? Fill in the values below that Ozobot speaks while the program is running. Are you expectations correct? Why do you think the title of this program is "Getting Dicey"? What happens to the values in the array elements each time you run the program with Ozobot on the Ozomap?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 9: One by One***

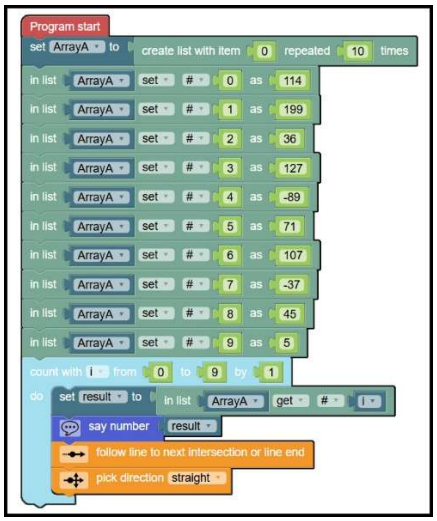


*Figure 11*

The program of Figure 11 has more blocks than all of the others in this lesson, but it is really not any more complex. The purpose of the ten *"in list set #"* blocks that begin the program is simply for the user to populate the array with whatever values are desired. What do you expect the values in each array element to be? Fill in the values below that Ozobot speaks while the program is running. Are your expectations correct?

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

***Program 10: An Improvement***

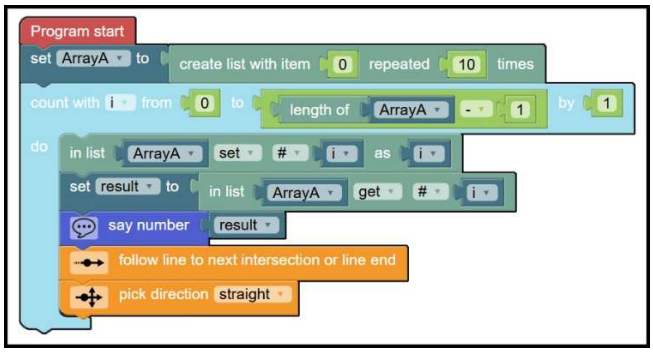


*Figure 12*

In Program 10 the output is the same as it was for the program of Figure 6. However, an important change has been introduced by including the *"length of"* block instead of hard coding the number 9 in the count with block, as was done in most of the previous programs. In this program the length of the array is declared as 10 in the *"set ArrayA to"* block. We then subtract 1 to get the index of the last element in the array. What is the main advantage of using the "get length of array block" here?

*Advantage:*_____

_____

***Program 11: Sort of Interesting***

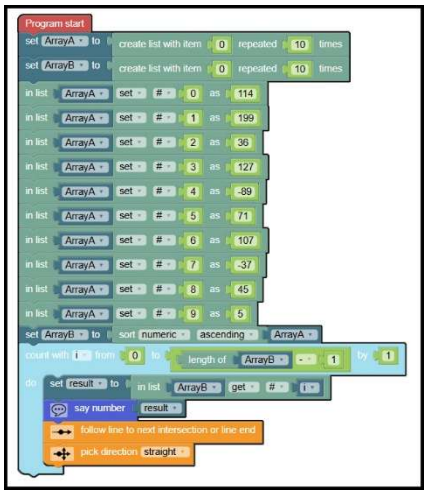


*Figure 13*

There are several differences between the program of Figure 13 and that of Figure 11:

1. A second array called ***ArrayB*** was declared with ten items all initialized to the value 0.
2. ***ArrayB*** was set to contain the values of ***ArrayA*** but sorted in ascending order.
3. The loop counter was adjusted to vary from 0 to the *length-1* of ***Array B***.
4. The variable ***result*** was changed to grab values from ***ArrayB*** rather than ***ArrayA***.

Make these changes to Program 9 and then run the modified program. What do you expect the values in each ***ArrayB*** element to be? Fill in the values below that Ozobot speaks while the program is running. Are your expectations correct?

| ArrayB[0] | ArrayB[1] | ArrayB[2] | ArrayB[3] | ArrayB[4] | ArrayB[5] | ArrayB[6] | ArrayB[7] | ArrayB[8] | ArrayB[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

What happens if you change from sort ascending to sort *descending*? Fill in the values below that Ozobot speaks while the program is running.

| ArrayB[0] | ArrayB[1] | ArrayB[2] | ArrayB[3] | ArrayB[4] | ArrayB[5] | ArrayB[6] | ArrayB[7] | ArrayB[8] | ArrayB[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

What happens if you now change the result line back to ***ArrayA***? Fill in the values below that Ozobot speaks while the program is running.

| ArrayA[0] | ArrayA[1] | ArrayA[2] | ArrayA[3] | ArrayA[4] | ArrayA[5] | ArrayA[6] | ArrayA[7] | ArrayA[8] | ArrayA[9] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

Has ***ArrayA*** been modified as a result of this sort? _____